An Improved Methodology for Automated Fault Tree Construction Using Component-Based Approach for Comprehensive System Modeling

氏名	マジダラ　アレフ　AREF MAJDARA
授与学位	博士（工学）
学位授与年月日	平成２１年９月９日
学位授与の根拠法規	学位規則第４条第１項
研究科，専攻の名称	東北大学大学院工学研究科（博士課程）技術社会システム専攻
学位論文題目	An Improved Methodology for Automated Fault Tree Construction Using Component-Based Approach for Comprehensive System Modeling (包括的システムモデル化のための要素ベースアプローチを用いた自動フォルトツリー作成の改良手法)
指導教員	東北大学教授　若林　利男
論文審査委員	主査　東北大学教授　若林　利男　東北大学教授　斉藤　浩海 東北大学教授　中田　俊彦　東北大学准教授　高橋　信

The major part of safety science is dedicated to identify different possible accidents and to take measures to prevent these accidents from happening or at least decrease the likelihood of occurrence of the accidents. Probabilistic Safety Assessment (PSA) methods are used to develop scenarios for hypothetical accidents, and to estimate the frequency of such accidents. Different PSA techniques have been developed, like Failure Mode and Effect Analysis (FMEA), Event Tree Analysis, and Fault Tree Analysis.

Fault Tree Analysis (FTA) is a very well-known and strong method for safety and reliability assessment. Fault tree is defined as a graphical representation of the various combinations of faults and failures that can result in an undesired event, called the Top Event. Historically, for the first time, fault tree analysis was used in 1961, by Bell Telephone Laboratory, as a technique to perform safety evaluation of Minuteman missile launch control system. Since then, there were significant improvements on mathematical and analytical techniques used in fault tree analysis; and today FTA has really wide applications in different fields of technology related to aerospace, nuclear, and chemical industries and it is opening its ways into many other areas such as robotics, rail transportation, and automotive industries.

However, manual construction of fault trees is a time consuming, error-prone and tedious task. Thus, there have been many attempts to get benefit of high speed and accuracy of computers to automate the process of constructing fault trees. Automated fault tree generation can be much easier and faster than the manual approach, and there will be fewer systematic errors.

There have been many studies performed in this field since 1970's, and various methods have been proposed by different researchers. Generally, an automated approach for fault tree construction can be considered as having two main phases: First, the system under study should be modeled in an appropriate way. This model should include all of the necessary details of the system, including hardware configuration as well as functional descriptions. The second

phase is an algorithm or procedure for generating the fault trees based on the model developed in the first step. In other words, the results of the first step are used as an input to the second phase.

The challenging part is mainly related to the first phase, which is finding an efficient modeling approach that can support modeling of different types of systems without ignoring any necessary details. The modeling methodology should be capable of generating the system model in reasonable time and with reasonable effort. Digraphs, Decision Tables, State Diagrams, Text Mining, etc. are some of the methods used in previous studies. Any of these methods have their own strong and weak points. For example, digraph-based approaches are very suitable for modeling continuous process plants and control loops, but modeling discrete parameters seems to be inconvenient in these methods, especially when some parameters have wide ranges of variation, which leads to very large digraphs, with lots of directed connections among the nodes. Semantic networks, introduced by Kumamoto and Henley are really strong in modeling various types of systems, as shown in their paper with several examples. However, preparing the semantic network itself is a time-consuming and difficult task, which can also increase the risk of human mistakes. Furthermore, both of the above-mentioned methods have no clear way of handling sequences of events. I have tried to get the useful ideas from the previous studies and construct a new methodology.

In this study I present a new method for computer-aided fault tree generation, which uses a component-based approach for system modeling and a trace-back algorithm for fault tree synthesis (Fig. 1). In this method, every system model is composed of some components and different types of flows flowing through them.

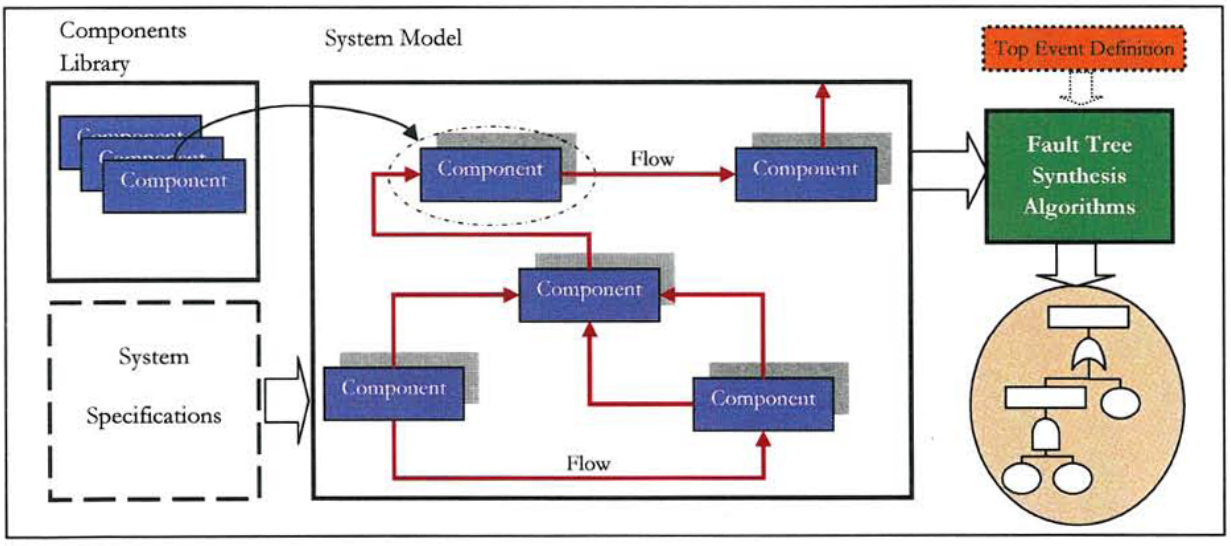

Fig. 1. Overview of the Proposed Approach.

Generally, a typical component model consists of a simple graphical symbol plus some functional descriptions. The basic symbol for a component is a simple box, having a caption, and several inward and outward arrows representing the input and output ports of the component, respectively. The component can be connected to other components of the system via these input and output ports. Also, each component has a *Function Table* that describes its input-output relations. In a function table, there is one column for any of the inputs and outputs, and also one column for

Functionality Condition, which is a parameter showing the internal condition of the component regarding its operability. It shows whether the component is working normally or it is in one of its failure modes.

For the components having different operational states, there is also a *State Transition Table*. This table shows how the component moves from one operational state to another, based on its inputs, functionality condition and the *commands* it receives from other components. For this type of components, a State column will be added to the function table and it affects the values of the component outputs.

A system model in this component-based method can be considered as a pipeline network through which different kinds of *Flows* are being transported from one component to another. In fact, the components of the system can communicate to each other by sending and receiving flows. So, in order to support different kinds of communications among components, I have defined four flow categories: Material Flow, Energy Flow, Information Flow, and Commands. These flows or their physical properties (like flow rate, temperature, pressure, etc.) appear in input or output columns of the function or state transition tables of the components.

In this component-based approach, any flow, flow property or other parameter must have a discrete (or descretized) value domain, so that it can be used in function tables or state transition tables. Choosing the appropriate domain value depends on the type of the parameter, system specifications, and the objectives of the analysis.

Basically, a component receives some inputs from other components of the system and performs some processing on the inputs, to create the outputs. Any flow that gets into a component will not necessarily leave the component from the other side, i.e. output ports. Some inputs are just necessary for the component to be able to operate, like the electricity input in the pump example, which is simply *consumed* inside the component, so that it can process the other input, which is the fluid to be pumped.

Human operators and external events are modeled as special components in this approach. A human operator is modeled as a component, with inputs, outputs, functionality conditions and any other parameter applicable to an ordinary component. Signal or data flows coming from alarms, indicators, control panels, etc. can be inputs to a human operator component. The outputs are usually, but not always, commands being issued to some other components, which can be physical devices or other personnel. Typical examples of external events include earthquakes, tornadoes, tsunamis and aircraft crashes. For being compatible with the component-based modeling approach, we model the external events as some special components. They have outputs which can enter some other components and affect their functionality. Both human operator and external event components have function tables to describe the way they function as a part of the system.

After constructing all the required component models, the system can be modeled by choosing desired components and making connections between them, according to the system specifications. Function tables and state transition tables of the components will simulate the way the system operates as a set of components. Flows in forms of materials, energy, etc. will be moving inside the system, from one component to another.

The proposed component-based approach is capable of modeling simple control loops. It is shown in this study how we can model the function of a control loop inside the function tables of the components constituting the loop.

Final step before starting to construct the fault trees is defining the top event. In this method, a top event is defined by assigning specific values to some of the system parameters, such as components' inputs, outputs, states, or some

parameters of a flow, like pressure, flow rate, or temperature. The values should be selected within the range of variation of the parameters.

The second phase of the automated process is constructing the fault tree by using the system model developed in the first part. The main part of the fault tree construction procedure is a trace-back algorithm to find all of the paths ending into the top event. When the top event is defined and all of the required settings are done, the trace-back algorithm can start from the point of occurrence of the top event, and examine the function tables and state transition tables of the components to find all of the potential causes of that top event. The algorithm moves from one component to another, in reverse directions, i.e. reverse to the directions of the flows.

When the algorithm is examining a component, its function table or state transition table is searched to find the rows with the output value of interest. Three different cases are possible: If there is a functionality condition of the component leading to that output value, a basic event is added to the tree and current trace-back path is terminated. However, if this is caused by the external inputs, trace-back is continued to the component from which these inputs are coming. Finally, in case of a *state* column affecting the output value for that row, the algorithm should jump to the corresponding state transition table. This procedure is continued until system boundaries are reached.

In order to show that the proposed approach can actually be utilized in practice, I have implemented the whole approach as a computer program, written in Delphi programming language. The program has an input interface for designing the components, a graphical interface for preparing the system model, and an output window for viewing the resultant fault trees in standard formats. Developing the computer program made it possible to test the different features of the approach, and get feedbacks for modifications and improvements.

After the program was developed and tested with several sample systems, I made two case studies for two actual systems: a part of an Unmanned Aerial Vehicle (UAV) system, and also a chemical plant. The case studies show the applicability of this methodology for real world systems. The results from my program are very close to the results from other studies, although there are some differences. The important point is that my approach identifies almost the same basic events contributing to the creation of the top event. In case of the UAV system, the results generated by the program revealed a human mistake in the manually constructed fault tree. The differences are mainly due to different algorithms used for fault tree construction. One obvious difference in the results is that my approach always includes a larger number of intermediate events. This is because in my method, fault tree synthesis is done in a component-by-component approach, and shows the entire fault paths in the fault tree. This might increase the calculation time, but, on the other hand, it gives more information about the exact fault propagation paths inside the system. It should be noted that for a specific top event defined for a system there is not a unique fault tree, so that it can be used as a reference for comparing the automatically generated results to it.

Future works on this subject can include creation of system-specific component libraries, containing ready to use component models, customized for being used in specific categories of systems. This can effectively reduce the time for computer-aided fault tree construction. More studies on modeling various control loop configurations can widen the range of applicability of the approach to more complex systems. Also, the proposed approach is open for further developments by studies on special concepts in fault tree analysis, like Common Cause Failures, Dynamic Fault Trees, and the use of other types of logic gates, like NOT, Priority AND (PAND), etc. in fault trees.

論文審査結果の要旨

確率論的安全評価において重要なフォルトツリー解析は、大規模科学技術システムの安全性や信頼性の評価にための有効な手法である。フォルトツリー解析は、航空、化学、原子力、交通分野等の多方面の分野で幅広く使用されてきている。いままで、計算機のスピードや正確性のメリットを利用したフォルトツリー作成プロセスの自動化について、多くの研究がなされてきた。自動フォルトツリー作成はマニュアルフォルトツリー作成に比べて、簡単で迅速にできるとともにシステマティクエラーを少なくできるというメリットを持っている。本研究は、システムのモデル化にコンポーネントベース手法を用い、更にフォルトツリー作成には Trace-back アルゴリズムを用いた新しい自動フォルトツリー作成手法を提案し、その有効性を明らかにしたものである。

本論文は、その成果をまとめたものであり、全文６章より構成される。

第１章は序論であり、本研究の背景、先行研究、目的について述べている。

第２章では、コンポーネントベースのシステムモデル化手法について述べている。対象とする科学技術システムのそれぞれのコンポーネントは、システム中の他のコンポーネントと結合された入力部と出力部で構成されている。システム内のそれぞれのコンポーネントは入出力の関係を記述した機能テーブルをもち、機能テーブルでは、コンポーネントの動作の成功、失敗を表している。このような手法により、システム構成の記述が容易にできる。

第３章では、Trace-back アルゴリズムを用いたフォルトツリー作成のためのシステムモデル化について述べられている。このアルゴリズムでは、起因事象を導くコンポーネントの失敗や失敗の組み合わせを見つけるまで、すべてのコンポーネントに対して事象が調べられる。

第４章では、提案手法を実際に使用できるようにするためのコンピュータープログラム化について述べている。本プログラムは、コンポーネントデザインのための入力インターフェース、フォルトツリー作成、グラフインターフェース等より構成されている。このプログラムにより熟練した専門家でなくとも容易にフォルトツリー作成が行えるようになった。

第５章では、４章で述べた自動フォルトツリー作成プログラムを用いて、実際のシステム（無人飛行機の電力供給システムと化学プラント）へ適用した結果について述べている。既にマニュアルで作成されたこれら２つのフォルトツリーと本プログラムで作成したフォルトツリー比較し、同等の結果が得られるとともに、フォルトツリー作成時間の短縮も明らかになり、新しい自動フォルトツリー作成手法の信頼性と有効性を明らかにしている。

第６章において、本研究の結論を述べている。

以上、本論文は、自動フォルトツリー作成の新しい手法を提案し、コンピュータープログラムの作成、実際のシステムへの適用により、その有効性を明らかにしたものである。本自動フォルトツリー作成システムは、従来、高度な専門性を必要していたフォルトツリー作成に対して、簡単で正確に、かつ、作成時間短縮を達成することができ、大規模科学技術システムの安全性や信頼性の評価に貢献するのみならず、リスク評価・管理学分野の発展に大いに寄与するものである。

よって、本論文は博士(工学)の学位論文として合格と認める。